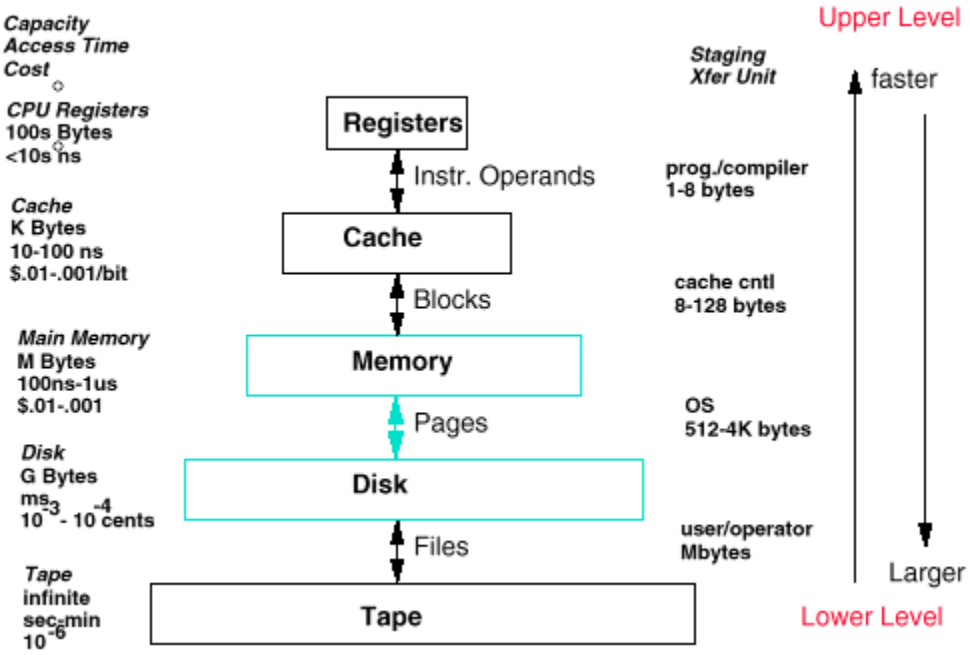
Cursul 8_2

Ierarhia in cadrul subsistemului de memorie



Performantele memoriei DRAM (*Exemplificare perioada 1984-1999*)

DRAM Generation						
1st Gen. Sample	'84	'87	'90	'93	'96	'99
Memory Size	1 Mb	4 Mb	16 Mb	64 Mb	256 Mb	1 Gb
Die Size (mm^2)	55	85	130	200	300	450
Memory Area (mm^2)	30	47	72	110	165	250
Memory Cell Area (μm^2)	28.84	11.1	4.26	1.64	0.61	0.23

(from Kazuhiro Sakashita, Mitsubishi)

Definitii reluare:

- **Hit time:** timpul de acces la blocul de la nivelul superior al ierarhiei de memorii, inclusiv timpul necesar deciziei de hit/miss
- **Miss penalty:** timpul necesar inlocuirii unui bloc aflat la nivelul superior al ierarhiei cu un bloc aflat la nivelul inferior si timpul de livrare a blocului catre procesor.

  Componentele lui Miss penalty:

  - **timpul de acces** = timpul necesar accesului la primul cuvant din bloc (este legat de latenta nivelului inferior;
  - **timpul de transfer** = timpul necesar transferului (depinde de largimea benzii)

In general o adresa de memorie specifica: adresa blocului (la nivelul ierarhic dat) si adresa cuvantului din bloc.

Adresa de memorie

Performanta ierarhiei de memorii:

Miss rate nu este potrivit pentru evaluarea performantei. Mai potrivit este indicatorul:

Timpul mediu de acces = Hit time * (1 – Miss rate) + Miss rate * Miss penalty

Acesta se poate masura in valori absolute (ns) sau in perioade de ceas de asteptare a Pc in cazul accesului la memorie.

Miss penalty, Miss rate si Timpul mediu de acces sunt functii de dimensiunea blocului (numarul de cuvinte din bloc)

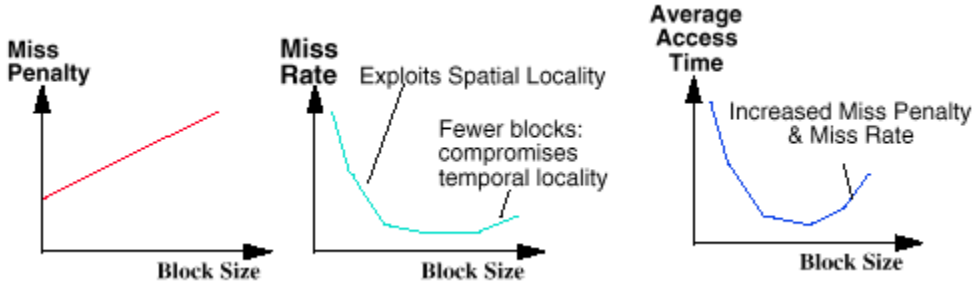
In general blocurile de dimensiuni mari se bucura de avantajul localitatii spatiale, cu urmatoarele observatii:

- Blocurile de dimensiuni mari conduc la o penalizare de absenta (miss penalty) mare: este necesar un timp mai mare pentru a incarca blocul
- Daca blocul are dimensiuni mari in raport cu dimensiunea memoriei intermediare, rata insucceselor (miss rate) va creste: vor fi prea putine blocuri in memoria intermediara;

In general,

Average Access Time = Hit Time x (1 - Miss Rate) + Miss Penalty x Miss Rate

Dependentele: Miss Penalty, Miss Rate si Average Acces Time ca functii de dimensiunea blocurilor



Probleme privitoare la ierarhia de memorii:

1. Unde poate fi plasat un bloc la nivelul superior al ierarhiei? ((Plasarea blocului)
2. Cum poate stabili daca un bloc se afla la nivelul superior al ierarhiei (Identificarea blocului);
3. Care bloc trebuie inlocuit la nivelul superior al ierarhiei in cazul unui insucces? (Inlocuirea blocului);
4. Ce se intampla la scriere? (Strategia de scriere)

Raspunsuri

1. Plasarea blocului in memoria intermediara (cache)

Actualmente se utilizeaza trei metode:

1.1. Maparea directa: fiecare bloc poate fi plasat intr-un singur loc in memoria intermediara.

Se considera ca memoria intermediara Mc si memoria principala Mp au capacitati, masurate in numar de blocuri, egale cu Nc si, respectiv Np (puteri ale lui 2). Numarul n, al blocului din Mc, in care va fi plasat un bloc, cu numarul p, din Mp, va rezulta din urmatorul calcul:

$$n = p \bmod (Nc)$$

n poate fi asociat cu adresa blocului in Mc

1.2. Asociativa: blocul din Mp se poate plasa in oricare pozitie/bloc din Mc

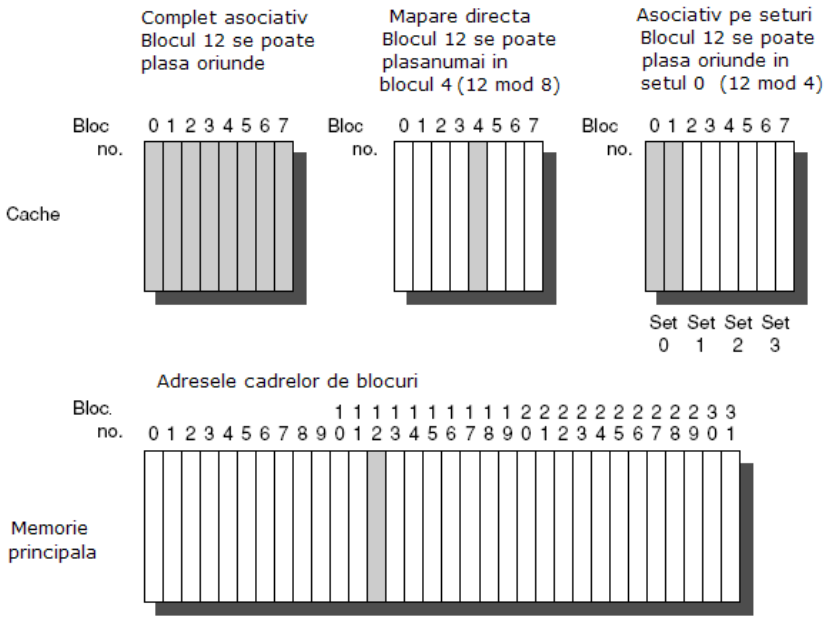
1.3. Asociativa pe seturi cu mai multe cai: blocul din Mp se poate plasa prin mapare directa numai intr-unul din cele Ns seturi de blocuri, create la nivelul lui Mc, prin gruparea a cate c seturi contigue in fiecare set; in cadrul setului plasarea se face asociativ.

Numarul s, al setului din Mc, in care va fi plasat un bloc, cu numarul p, din Mp, va rezulta din urmatorul calcul:

$$s = p \bmod (Ns)$$

Exemple: Se considera o memorie Mc cu o capacitate de 8 blocuri si o memorie Mp cu o capacitate de 31 de blocuri.

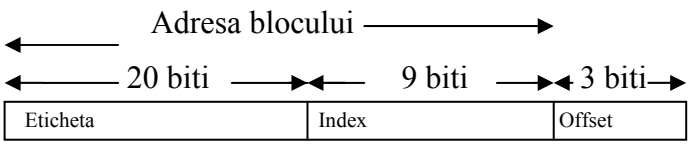
- In cazul maparii directe blocul din Mp, cu numarul p =12, va fi plasat in blocul cu numarul n = 12 mod 8 = 4, din Mc.
- In conditiile amplasarii asociative, blocul cu numarul 12, din Mp, poate fi plasat in oricare dintre cele 8 blocuri din Mc.
- In situatia amplasarii asociative pe seturi cu c cai (c =2, => Ns = 4)), numarul setului s, in care va fi plasat blocul din Mp, cu numarul p = 12, va fi: s = 12 mod 4 = 0.



Memoria cache are 8 blocuri iar memoria principala are 32 de blocuri

2. **Cum poate stabili daca un bloc se afla la nivelul superior al ierarhiei** (Identificarea blocului in memoria intermediara)?

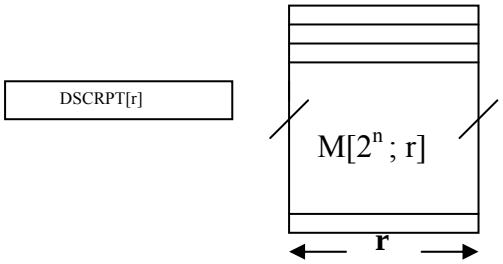
2.1.Adresa furnizata de catre procesor are urmatoarea structura (VAX 11/780, Mc asociativa pe doua cai, cu doua cuvinte pe bloc):



- Offset-ul indica adresa/pozitia cuvantului sau octetului in cadrul blocului;
- Indexul ofera o indicatie asupra capacitatii Mc, ca numar de blocuri sau de seturi asociative;
- Eticheta da o indicatie asupra numarului blocului plasat in Mc.

2.2. Unitatea logica combinationala ASOC

Aceasta unitate logica combinationala compara continutul unui registru "descriptor" – DSCRPT[r], simultan cu continuturile a 2^n celule de memorie (M) si furnizeaza un vector cu 2^n ranguri, dintre care cel mult un rang j este egal cu 1, celelalte fiind egale cu 0. Rangul j ia valoarea 1 daca continutul locatiei M[j] este identic cu continutul descriptorului



UNIT: ASOC(M;DSCRPT)

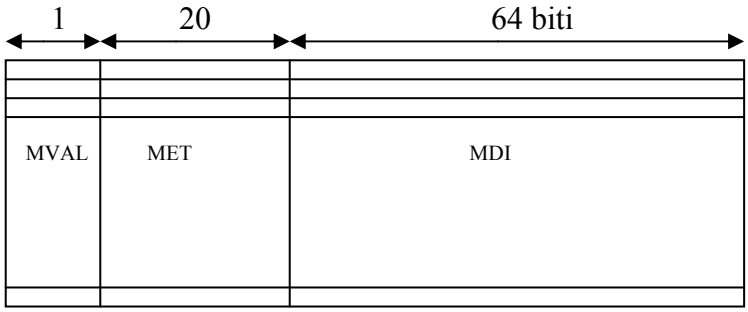
INPUTS: $M[2^n ;r]$; DSCRPT[r]

OUTPUTS: $ASOC[2^n]$

1. j = 0
2. $ASOCj = \overline{\cup / (M[j] \oplus DSCRPT)}$
3. j <= j + 1
4. $=>(j < 2^n)/(2)$

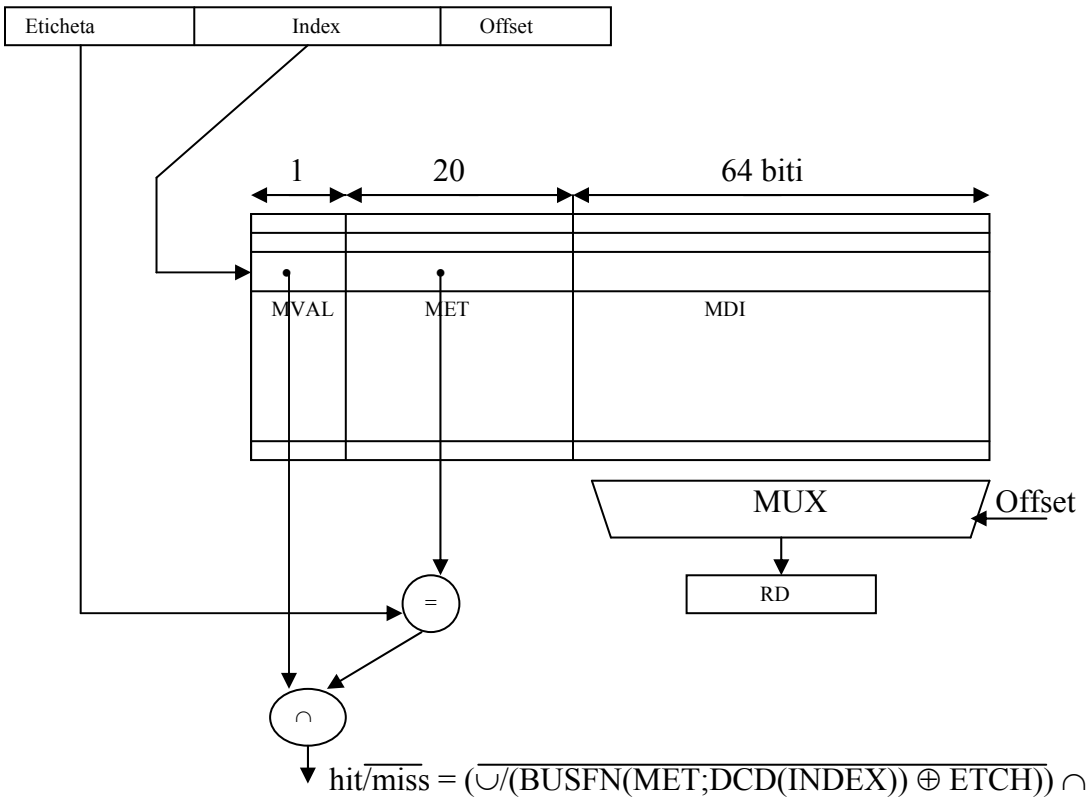
END

2.3. Memoria intermediara contine, pe langa memoria de date/instructiuni (MDI), o memorie pentru etichete (MET) si o memorie pentru informatia de validitate (MVAL) a informatiei din MDI. Fiecarui cuvant din MDI ii sunt atasate un cuvant din MET si un cuvant, de 1 bit, din MVAL



2.4. Pentru a stabili daca un bloc se afla in Mc trebuie, in functie de metoda de amplasare, sa se genereze, pe cale logica, informatia de hit/miss

2.4.1. In cazul maparii directe se poate folosi schema de mai jos:



$$\overline{hit/miss} = (\cup/(\overline{BUSFN(MET;DCD(INDEX)) \oplus ETCH)}) \cap BUSFN(MVAL;DCD(INDEX))$$

Citire:

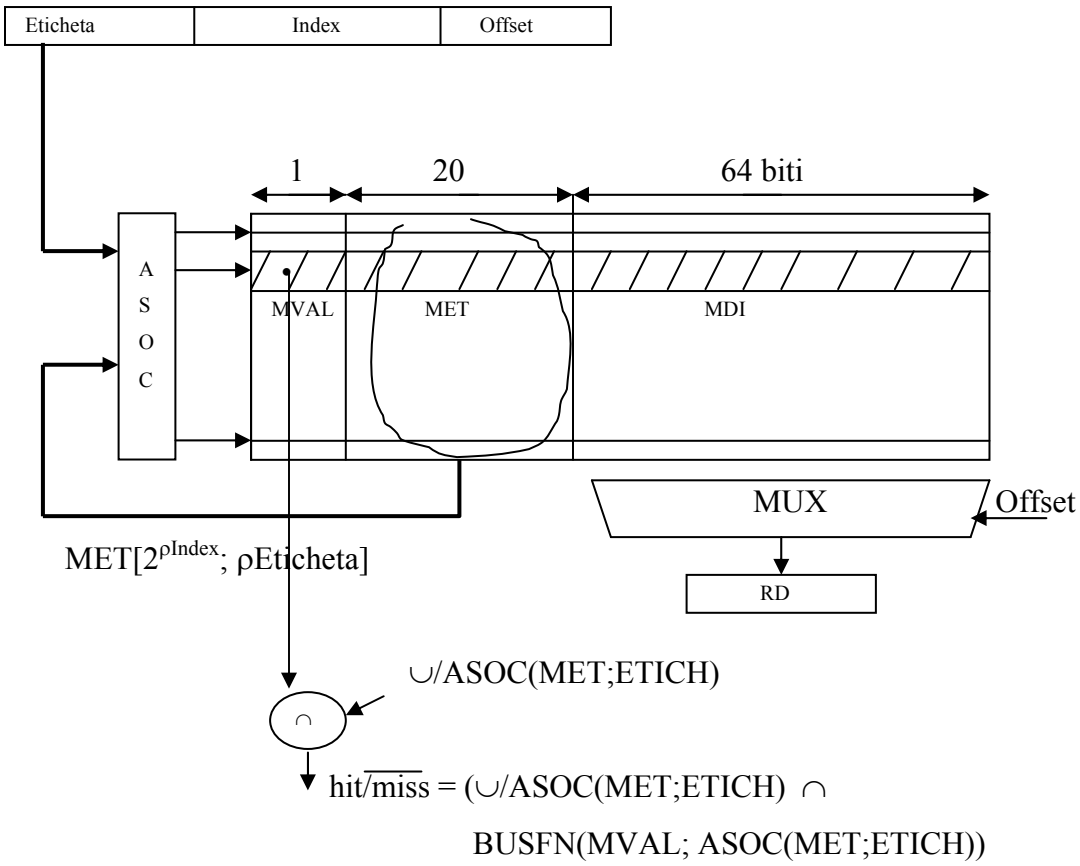
RD ← BUSFN(MDI;DCD(INDEX)) * hit /* fara Offset

Scriere:

MDI * DCD(INDEX) ←RD*hit

MVAL * DCD(INDEX) ← 1

2.4.2. In cazul amplasarii asociative se poate folosi schema de mai jos:



Citire:

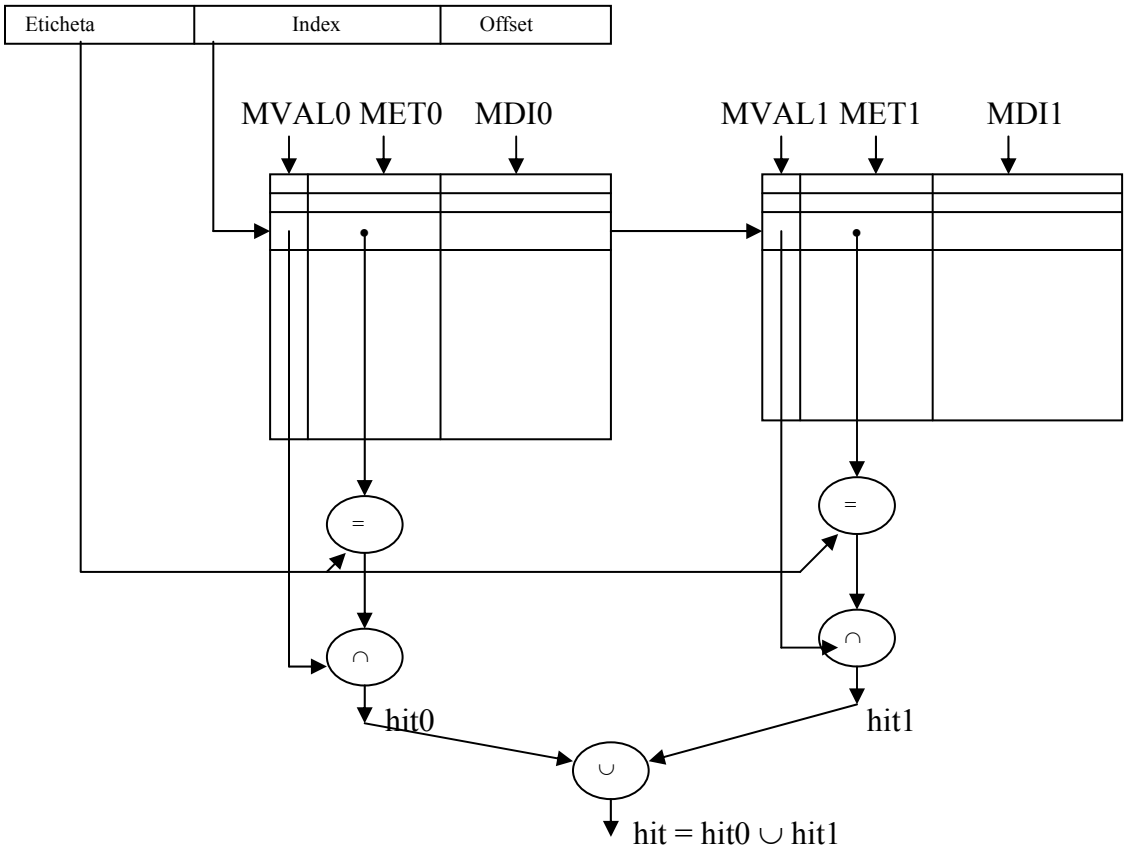
RD←BUSFN(MDI; ASOC(MET;ETICH)) ∩ hit /* nu se considera offset-ul

Scriere:

MDI * ASOC(MET;ETICH)←RD *hit

MVAL * ASOC(MET;ETICH)) ←hit

2.4.3. In cazul amplasarii asociative pe seturi cu doua cai se poate utiliza schema urmatoare:



hit0 = (∪/(BUSFN(MET0;DCD(INDEX)) ⊕ ETCH)) ∩
BUSFN(MVAL0;DCD(INDEX))

hit0 = (∪/(BUSFN(MET1;DCD(INDEX)) ⊕ ETCH)) ∩
BUSFN(MVAL1;DCD(INDEX))

Citeste:

RD←(BUSFN(MDI0;DCD(INDEX)) ! BUSFN(MDI1;DCD(INDEX))) *(hit0,hit1)

Scrie:

(MDI0*DCD(INDEX)!MDI1*DCD(INDEX))*(hit0,hit1) <- RD

(MVAL0*DCD(INDEX)!MVAL1*DCD(INDEX))*(hit0,hit1) <- 1

3. Care bloc trebuie inlocuit la nivelul superior al ierarhiei in cazul unui insucces? (Inlocuirea blocului);

3.1. Cazul maparii directe nu ridica probleme.

3.2. Cazurile asociativ pe seturi si complet asociativ.

3.2.1. Blocurile candidate la inlocuire se selecteaza aleator. Unele sisteme folosesc scheme pentru dispersarea datelor pe un set de blocuri, intr-o maniera pseudoaleatoare, pentru a obtine o comportare reproductibila.

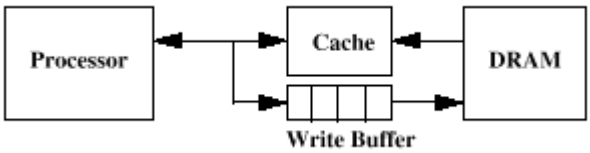
3.2.2. Cel mai putin recent utilizat bloc (LRU - Least Recently Used), presupune utilizarea unor mecanisme specifice, de exemplu: contor de utilizare pentru fiecare bloc.

Associativity:	2-way		4-way		8-way	
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

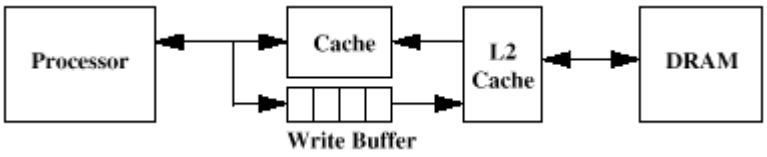
4. Strategia la scriere.

- *Write through*—Informatia este scrisa ,atat in blocul din memoria intermediara, cat si in blocul din memoria de la urmatorul nivel
- *Write back*— Informatia este scrisa numai in blocul din memoria intermediara. Continutul blocului modificat este scris in memoria principala numai in cazul inlocuirii lui in memoria intermediara. Blocul poate fi curat sau murdar (modificat/scris)
- Avantajele/dezavantajele pentru WT si WB:
    - WT: insuccesele la citire nu conduc la scrieri, usor de implementat, necesita banda mare la interfata intre memoriile intermediara si principala, nivelul inferior contine informatia actualizata (pentru I/E, alte procesoare).

- WB: nu apar scrieri pentru scrieri repetate; necesita banda redusa.WT este intotdeauna combinat cu tampoane/buffere pentru scriere, astfel nu se asteapta din cauza memoriei de la nivelul imediat inferior



- Tamponul/Buffer-ul plasat intre memoria intermediara si memoria principala
  - Procesoriul scrie datele in memoria intermediara si in tampon
  - Controlorul memoriei scrie continutul tamponului in memorie
- Tamponul de scriere este FIFO
  - Numarul tipic de intrari: 4
  - Opereaza bine daca frecventa memoriei (w.r.t. time) << 1/DRAM (write cycle)
- Dificultatile proiectantului sistemului de memorie:
  - frecventa memoriei (w.r.t. time) > 1/DRAM (write cycle)
  - saturarea tamponului de scriere
- Daca conditia: (w.r.t. time) > 1/DRAM (write cycle) se mentine mai mult timp (Ciclul UCP prea mic (rapid) si/sau exista prea multe instructiuni de memorare in secventa:
  - Tamponul de scriere se va satura indiferent de capacitateqa lui
  - CPU Cycle Time <= DRAM Write Cycle Time
- Solutia pentru cazul saturarii tamponului de scriere
  - Utilizeaza metoda WB;
  - Instaleaza nivelul2 de memorie intermediara (L2 cache):



Pentru insuccese/misses la scriere apar doua situatii:

- Blocul este incarcat din memoria de la nivelul inferior, dupa care se scrie in el (Write Allocate or Fetch on Write). Presupune utilizarea in continuare a blocului.
- Blocul este modificat la nivelul inferior, fara sa fie adus in memoria intermediara (Write No Allocate or Write Around)

Exemplu: Se presupune ca o scriere pe 16 biti la locatia de memorie 0 x 0 conduce la insucces. Trebuie sa se aduca un bloc?

- DA: Write Allocate.
- NU: Write Not Allocate